中文信息处理 Chinese Information Processing

第六章 作业

14307130318 刘超颖

1. 使用任何本章所述的三种分类器之一，以及你能想到的特征，建立一个名字性别分类器。从将名字语料库分成3 个子集开始：400 个词为测试集，400个词为开发集，剩余的个词为训练集。然后从示例的名字性别分类器开始，逐步改善。使用开发集检查你的进展。一旦你对你的分类器感到满意，在测试集上检查它的最终性能。相比在开发测试集上的性能，它在测试集上的性能如何?

使用朴素贝叶斯分类器。

首先，我们对书中的特征进行测试，即以名字末位字母为特征，因为我们发现：以a，e 和i结尾的很可能是女性，而以k，o，r，s结尾的很可能是男性。

```
>>> import nltk
>>> def gender_features(word):
...     return {'last_letter': word[-1]}
...
>>> from nltk.corpus import names
>>> import random
>>> names = ([(name, 'male') for name in names.words('male.txt')] +
[(name, 'female') for name in names.words('female.txt')])
>>> random.shuffle(names)
>>> train_names = names[800:]
>>> test_names = names[:400]
>>> devtest_names = names[400:800]
>>> train_set = [(gender_features(n), g) for (n,g) in train_names]
>>> devtest_set = [(gender_features(n), g) for (n,g) in devtest_names]
>>> test_set = [(gender_features(n), g) for (n,g) in test_names]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
>>> print(nltk.classify.accuracy(classifier, devtest_set))
>>> print(nltk.classify.accuracy(classifier, test_set))
```

其性能并不是很好，如下图：

```
中文信息处理 — python — 80×24
Python 2.7.10 (default, Jul 30 2016, 18:31:42)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> def gender_features(word):
...     return {'last_letter': word[-1]}
...
>>> from nltk.corpus import names
>>> import random
>>> names = ([(name, 'male') for name in names.words('male.txt')] + [(name, 'fem
ale') for name in names.words('female.txt')])
>>> random.shuffle(names)
>>> train_names = names[800:]
>>> test_names = names[:400]
>>> devtest_names = names[400:800]
>>> train_set = [(gender_features(n), g) for (n,g) in train_names]
>>> devtest_set = [(gender_features(n), g) for (n,g) in devtest_names]
>>> test_set = [(gender_features(n), g) for (n,g) in test_names]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
>>> print(nltk.classify.accuracy(classifier, devtest_set))
0.7325
>>> print(nltk.classify.accuracy(classifier, test_set))
0.7675
>>>
```

通过对英文名特征的观察，我们发现，名字末两位以cy，va和na结尾的很可能是女性，而以ck，er，el结尾的很可能是男性。所以我们改以名字末两位字母为特征进行分类。

```python
import nltk

def gender_features(word):
    return {'first_letter': word[-2:]}

from nltk.corpus import names
import random

def process():
    namelist = ([(name, 'male') for name in names.words('male.txt'
)] + [(name, 'female') for name in names.words('female.txt')])
    random.shuffle(namelist)
    train_names = namelist[800:]
    test_names = namelist[:400]
    devtest_names = namelist[400:800]
    train_set = [(gender_features(n), g) for (n,g) in train_names]
    devtest_set = [(gender_features(n), g) for (n,g) in devtest_na
mes]
    test_set = [(gender_features(n), g) for (n,g) in test_names]
    classifier = nltk.NaiveBayesClassifier.train(train_set)
    print(nltk.classify.accuracy(classifier, devtest_set))
    print(nltk.classify.accuracy(classifier, test_set))
```

```
>>> import gender
>>> gender.process()
```

```
中文信息处理 — python — 80×24
LCYmengmengdadeMacBook-Pro:中文信息处理 LCY$ python
Python 2.7.10 (default, Jul 30 2016, 18:31:42)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import gender
>>> gender.process()
0.7575
0.775
>>>
```

可以看出，将特征改为名字末两位字母后，性能有一定提升。
相比在开发测试集上的性能，它在测试集上的性能往往更好。

2. 使用本章讨论过的电影评论文档分类器，产生对分类器最有信息量的20个特征的列表。你能解释为什么这些特定特征具有信息量吗？

```
>>> import nltk
>>> from nltk.corpus import movie_reviews
>>> documents = [(list(movie_reviews.words(fileid)), category) for
category in movie_reviews.categories() for fileid in
movie_reviews.fileids(category)]
>>> import random
>>> random.shuffle(documents)

>>> def document_features(document):
...     document_words = set(document)
...     features = {}
...     for word in word_features:
...             features['contains(%s)' % word] = (word in
document_words)
...     return features
...
>>> all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
>>> word_features = all_words.keys()[:2000]
>>> featuresets = [(document_features(d), c) for (d,c) in documents]
>>> train_set, test_set = featuresets[100:], featuresets[:100]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
>>> classifier.show_most_informative_features(20)
```

```
中文信息处理 — python — 80×24
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
>>> classifier.show_most_informative_features(20)
Most Informative Features
           contains(sans) = True              neg : pos    =     10.0 : 1.0
     contains(mediocrity) = True              neg : pos    =      8.5 : 1.0
          contains(wires) = True              neg : pos    =      7.0 : 1.0
           contains(hugo) = True              pos : neg    =      6.9 : 1.0
      contains(dismissed) = True              pos : neg    =      6.3 : 1.0
    contains(bruckheimer) = True              neg : pos    =      6.3 : 1.0
         contains(fabric) = True              pos : neg    =      5.7 : 1.0
    contains(overwhelmed) = True              pos : neg    =      5.7 : 1.0
    contains(understands) = True              pos : neg    =      5.6 : 1.0
            contains(ugh) = True              neg : pos    =      5.6 : 1.0
      contains(uplifting) = True              pos : neg    =      5.5 : 1.0
         contains(doubts) = True              pos : neg    =      5.2 : 1.0
          contains(tripe) = True              neg : pos    =      5.1 : 1.0
   contains(accomplishes) = True              pos : neg    =      5.1 : 1.0
        contains(topping) = True              pos : neg    =      5.1 : 1.0
           contains(wits) = True              pos : neg    =      5.1 : 1.0
           contains(lang) = True              pos : neg    =      5.1 : 1.0
   contains(effortlessly) = True              pos : neg    =      5.0 : 1.0
        contains(quicker) = True              neg : pos    =      4.8 : 1.0
          contains(locks) = True              neg : pos    =      4.8 : 1.0
>>>
```

因为这些特定特征大多是具有明显情感特征的形容词。